# Computing education in children's early years

# *Computing education in children's early years: a call for debate*


## Abstract

International changes in policy and curricula (notably recent developments in England) have led to a focus on the role of computing education in the early years. As interest in the potential of computing education has increased, there has been a proliferation of programming tools designed for young children. While these changes are broadly to be welcomed, the pace of change has arguably led to more attention to the tools than to key questions about pedagogy.

This paper proposes three areas of research (Logo, computational thinking, and teaching STEM subjects of science, technology, engineering and mathematics) that may inform computing education for young children and suggests that a greater focus on thinking skills and connections to manifestations of computers in the real world is needed. Above all, the paper calls for an informed debate about the trend towards introducing computing education to children in the early years.


## Structured practitioner notes

What is already known about this topic

- Computing education has been recognised as an important area of learning.
- Programming plays an important role in computing education.
- An increasing number of tools are now available to support the teaching of programming to children in the early years.

What this paper adds

- A consideration of the conceptual and pedagogical implications of computing education for young children.
- A call to inform computing education by drawing on earlier research on Logo, computational thinking and teaching STEM subjects in the early years.
- A statement of the need to consider the conceptual thinking that underlies computing education as well as to evaluate the benefits of tools for supporting programming skills.

Implications for practice and/or policy

- Teachers may benefit from greater support in understanding the foundations of computing and its relation to other learning.
- Pedagogy needs to take account of the social and affective aspects of computing education and how it relates to children's everyday experiences in the world.
- With the benefit of opportunities for professional learning, teachers could develop the skills to evaluate programming tools or create their own activities to support computing education.

## Introduction

The trend towards computing education for children in the early years presents an exciting opportunity to tap into children's early potential for learning, especially if it takes place before some of the gender stereotypes associated with computing (Robertson, 2013; Wajcman, 2007) begin to influence their choices. Children in the UK start school at about the age of five, and in this paper we refer to the 'early years' as the period immediately before and after this transition, including children between the ages of three and six.

There is currently no clear consensus on what we mean by computing education in relation to young children or on what is an appropriate pedagogy in the early years (Cooper, Bookey, & GruenBaum, 2014). The introduction of computing education on a wide scale in England[1], combined with educators' lack of confidence in this area of the curriculum and limited opportunities for professional learning (Brown, Sentance, Crick, & Humphreys, 2014) has been met with the increasing availability of a number of tools that appear to offer solutions. However, notwithstanding noteworthy efforts such as CS Unplugged, the dominance of tools that emphasise programming skills rather than a deeper conceptual understanding has implications for the role of the teacher and for pedagogy, conceptualised here as the art and craft of teaching (Eisner, 1983). The paper scrutinises various terms relating to computing education in the early years and, in doing so, suggests that greater consideration of previous literature in the field may help renewed attempts to bridge research and practice.

*Definitions*

Discussions of computing education often include related terms, such as computer science, programming, coding, algorithms, or computational thinking. These terms are not synonymous, so defining what we mean by them can help to clarify the extent to which different skills or concepts are appropriate for young children.

In an educational context, one definition of computing provided by the UK-wide organisation Computing at School is that it is "concerned with how computers and computer systems work and how they are designed and programmed" (Berry, 2013, p.4). This definition emphasises programming, yet it is not always clear what exactly is meant by the term programming, nor the extent to which computing requires procedural competence as opposed to a more conceptual understanding. Although the relationship between conceptual and procedural knowledge is likely to be iterative (Rittle-Johnson, Siegler, & Alibali, 1999), their relative importance and how they might be achieved are important considerations when thinking about pedagogy.

Out-of-school organisations, such as Code Club and CoderDojo, in which coding is a central activity, have recently come to prominence. This raises the issue of the relationship between coding and programming. For the purposes of this discussion, we consider *coding* to refer to the specific skills of inputting instructions using a particular language, such as Java or Scratch, whereas *programming* reflects the wider design and implementation process of using code to solve particular problems.

## Computing education in the early years

In the UK, the need to transform computing education was promoted by *Shut Down or Restart*, a report by the Royal Society (2012) that aimed to alert practitioners and policymakers to the need for a highly skilled workforce if the UK is to remain competitive. The report argued for a shift in the school curriculum approach to ICT from the existing focus on training to use suites of office-oriented software to

computing. According to the report, computing is "concerned both with computers and computer systems – how they work and how they are designed, constructed, and used – and with the underlying science of information and computation" (p.5). Computing education encompasses computer science, defined as "[t]he rigorous academic discipline, encompassing programming languages, data structures, algorithms, etc." (p.5).

Evidently, programming is a key feature of computing in the Royal Society report and this is particularly noticeable in its only reference to the early years. Here, it both reinforces and qualifies this position by describing how children can gain direct experience of programming ideas "albeit using relatively simple control interfaces to programmable toys" (p.48) and providing an example of the Bee-bot, a simplified floor turtle.

*Computing in the national curriculum for England*

The Royal Society report was influential in prompting change and Computing was introduced, with some speed, as a subject in the national curriculum for England from September 2014. Computing now begins when children start school, with key stage 1 covering the first two years from ages five to seven. The curriculum document (Department for Education, 2013) itemises six main areas to be taught:

- understand what algorithms are; how they are implemented as programs on digital devices; and that programs execute by following precise and unambiguous instructions
- create and debug simple programs
- use logical reasoning to predict the behaviour of simple programs
- use technology purposefully to create, organise, store, manipulate and retrieve digital content
- recognise common uses of information technology beyond school
- use technology safely and respectfully, keeping personal information private; identify where to go for help and support when they have concerns about content or contact on the internet or other online technologies.

The first three of these objectives make explicit reference to programs so it is unsurprising that many commercial programming tools are now promoted as helping to address these curriculum targets. We mention several such resources in the course of this discussion and more information may be found through the web links provided at the end of the paper.

Out-of-school clubs are less constrained in their activities than schools. Possibly as a result of the demands of the learning tools used, clubs such as these generally cater for slightly older children, their websites indicating that Code Club is for children aged 9-11 and CoderDojo for those aged 7 to 17. Whether such organisations assume that younger children are not able to cope with the conceptual demands of programming or whether their emphasis on older children is based on pragmatic issues such as the lower teacher-child ratio required for this age range or the lack of qualified staff in early years education is not clear.

*Significance of an early years focus*

The marketing of tools for younger children, along with curriculum targets from the start of school, demonstrate the need to consider computing education from age three or four.

Calls for computing for all (Wing, 2006) raise questions about whether or not there exists an appropriate age to introduce children to computing. In England, the computing curriculum starts upon entry to school at age five. Currently, it remains unclear how moves toward computing have been integrated into the Early Years Foundation Stage in England for children aged three to five, although various unofficial efforts have been made to propose activities (e.g. iCompute) Supporting children in pre-school settings raises questions about progression in subsequent years. For instance, the early introduction of programming floor robots may underpin later learning or may simply foreshadow later school-based activities with the concomitant risk of repetition.

Problems relating to the progression of learning outcomes and activities are also evident in other curricula such as the Scottish Curriculum for Excellence. In the section of Technologies: Experiences and Outcomes on 'technological developments in society' there is some differentiation between "I enjoy playing with and exploring technologies to discover what they can do and how they can help us" (Early phase) and "By exploring and using technologies in the wider world, I can consider the ways in which they help" (First phase), although it is difficult to see how educators would operationalise this. However, the early and First phases are elided in the section on 'Computing science contexts for developing technological skills and knowledge', with undifferentiated guidance given as "I am developing problem-solving strategies, navigation and co-ordination skills as I play and learn with electronic games, remote control or programmable toys" (Education Scotland, 2009).

While many of the issues raised by early computing are equally valid in the context of older children, we indicate here some reasons to give special consideration to this age group.

Educational impact

Research has shown that children's ability in mathematics at age five is a significant predictor of later ability (Sylva, Melhuish, Sammons, Siraj-Blatchford, & Taggart, 2009). This impact of early learning on later achievement (Feinstein, Duckworth and Sabates 2008) means there is a case for getting computer education right and ensuring that children have equal access to opportunities to learn, both at school and home.

Pedagogy

Rather than being seen simply as less competent versions of older children, young children have particular learning needs that should be reflected in appropriate pedagogical approaches. Depending on the national context, educators in the early years subscribe to the view that play is an important medium for learning and tend to adopt a cross-curriculum approach that recognises the physical, cognitive, linguistic and social and emotional aspects of learning (Plowman & Stephen, 2005). Tools therefore need to be designed in such a way that they respect early years pedagogy.

Role of the adult

When children are still developing as independent learners it is important that the educator feels knowledgeable and confident. As parents and other caregivers, as well

as practitioners in the kindergarten or school, fulfil the role of educator it also needs to be recognised that these adults are unlikely to have experiences of computing from their own education or career that they can draw upon.

Cost

Many of the tools required to develop computing in the early years have cost implications. Equipment, such as programmable toys, may be expensive and while software may be free it may only function on particular devices, such as iPads. The importance of the home as a site of learning in the early years means that variations in availability of devices across homes and schools may also impact on opportunities.

Gender

There are widespread concerns over the lack of women in computer science (Robertson, 2013) and how to address this imbalance. Focus on the early years has the potential to engage both boys and girls before stereotypes develop about their suitability for learning how to program.

## Research in early years computing

The relative recency of the trend towards computing in the early years may imply a dearth of research available to inform this area of learning. Yet this ignores the wealth of literature that is relevant to computing education in the early years, but may have been published more than a decade ago, or in other disciplines such as mathematics. In this section, therefore, we consider relevant research that was published in the past as well as more recent work.

Although attention has been paid to the history of computing education in regard to recent developments in schools (Grover & Pea, 2013), it is unusual to focus on the early years. In this section, therefore, we consider three areas of research that are particularly relevant. The first is research based upon Logo, which offers an historical perspective on efforts to teach computing. The second relates this earlier work to more recent literature on the concept of Computational Thinking. The third area, STEM (science, technology, engineering and mathematics) education in the early years, emphasizes the significance of work in other domains.

### *Logo*

Recent debates about how to teach computing to children often neglect the wealth of research from the 1980s and 1990s focused on tools such as the BBC Micro and Logo. Developed in the late 1960s by a team including Seymour Papert, Logo is a simplified programming language that was originally designed as part of an experiment to test the idea that children can learn through programming (Layman & Hall, 1988). In its early days, the most popular use of Logo involved writing code to direct the movements of physical writing device: “a floor turtle”. With the proliferation of personal computers in the late 1970s, this physical device moved to become an on-screen turtle graphic (Sargent, Resnick, Martin, & Silverman, 1996).

Although the tool was not designed to teach computing so much as mathematical and logical problem-solving skills, Logo’s simplicity provided children with a window into the power of computing and reminds us that programming may be considered a means to immerse children in conceptually challenging problem solving. In this regard, there are noteworthy similarities between the logical problem solving associated with Logo and what has recently been defined as ‘algorithmic problem

solving' (Barr & Stephenson, 2011) but this term may seem overly specialised for practitioners and interpreted as too remote from classroom practice.

As Logo was used widely in classrooms for a prolonged period, its use attracted research that attempted to address questions that continue to be valid today. One central question was whether programming with Logo transferred to other areas of learning. Research results were mixed: while some studies showed beneficial cognitive effects (see Clements & Gullo, 1984 for a review), the criticism remained that learning with Logo did not transfer to more general thinking skills (Pea, 1983).

According to Papert, studies focusing on the Logo effect were 'technocentric' (Papert, 1987) and misunderstood the theoretical perspective underpinning Logo as a powerful means for children to express, share, reflect and develop thinking. However, in the years that followed its introduction in the 1980s, concerns were expressed that there was a lack of training for teachers to understand and integrate this deeper theoretical approach (Sutherland, 1993). As seen in Brown et al (2014) these concerns persist, even if they refer to more recent tools.

*Computational thinking*

In 2006, Jeanette Wing presented a paper emphasising the need for all individuals, not just computer scientists, to be able to think computationally as an everyday life-skill, later defining computational thinking as "the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" (Wing, 2011). The term builds on Papert's earlier work and has stimulated debates around how to define, teach and assess computing (Grover & Pea, 2013).

Wing's definition points to the centrality not only of problem *solving*, but also formulating problems or problem *finding*. Both these aspects are familiar to innovative early years pedagogy in other STEM subjects (Cheung, 2013). The definition also describes the need to represent solutions in ways that can be understood not only by computers, but also by humans. The inclusion of humans as 'information processing agents' broadens the notion of computational thinking to allow for a graduation from general communication skills to the specific practice of coding and allows for familiar activities such as giving the robot instructions to make a jam sandwich.

Wing's efforts to broaden the remit of computing by emphasising thinking skills presents an opportunity to frame computing as accessible to early years pedagogy but it is not clear how far this has filtered down into practice. Wing herself refers to the need for expertise from educators to make this link and highlights the importance of its early introduction: "If we wanted to ensure a common and solid basis of understanding and applying computational thinking for all, then this learning should best be done in the early years of childhood" (Wing, 2008, p.3720).

*STEM education in the early years*

Research into learning in the early years emphasises the importance of the social and affective and this child-centredness has influenced thinking about pedagogy. Teachers have a particular expertise in orchestrating activities that motivate children by focusing on their interests, encouraging children to work collaboratively and exploiting their creativity. According to findings from the Creative Little Scientist project, creativity in STEM subjects can be defined as "generating ideas and strategies

as an individual or community, reasoning critically between these and producing plausible explanations and strategies consistent with the available evidence" (Compton et al., 2014, p.4).

By engaging children in thinking about real-world examples such as driverless cars and self-service shop tills there are opportunities for them to explore, experiment and reflect upon different ideas collaboratively. As such, it is possible to draw links between computing and what is thought of traditionally as early science education. Indeed, the Royal Society's definition of computational thinking as "the process of recognising aspects of computation in the world that surrounds us, and applying tools and techniques from computer science to understand and reason about both natural and artificial systems and processes" (Furber, 2012, p. 29) makes this link and suggests that it could be fruitful to apply some of the principles associated with STEM education in the early years to early computing.

## Computing education in the early years: the way forward?

An agenda similar to that promoted by the Royal Society's (2012) *Shut Down or Restart* has also gained traction in the United States with *Running on Empty: The Failure to Teach K–12 Computer Science in the Digital Age* (Wilson, Sudol, Stephenson, & Stehlik, 2010) influencing debate. According to Barr and Stephenson (2011), the successful integration of computational thinking concepts into the curriculum requires a change in educational policy and K-12 teachers need resources, "starting with a cogent definition and relevant age-appropriate examples". In their review of computational thinking, Grover and Pea (2013) argue that current energies have moved from a focus on definitions and learning environments to thinking about how to promote and assess computational thinking.

Fessakis, Gouli and Mavroud (2013) suggest that it is not the lack of tools that hinders progress in computing education but the development of appropriately designed learning activities and supporting material which can be easily integrated in every day school practice by "well informed and prepared teachers". In England, various efforts to support teachers with the new curriculum are visible, not least *The Barefoot Project* funded by the Department of Education. The website provides valuable supporting materials and exemplars for teachers, although there is an understandable emphasis on specific programming tools with which to address curriculum objectives.

### *Using programming tools to explore learning*

Innovative designs may have the potential to lower the age threshold at which young children can engage meaningfully with programming. Perlman's (1976) early Tortis slot machine that controls a turtle with physical commands cards or the tangible programming blocks (e.g. Wyeth & Wyeth, 2001) are examples of what was seen previously as accessible forms of interface. More recently, we have seen a proliferation of new resources (e.g. Hopscotch, Kodable, Lightbot and Tynker) as designers have taken advantage of the touchscreen interaction offered through tablet devices such as the iPad. As documented elsewhere (Manches, 2013), the more direct interaction afforded by the interfaces may make them suitable for children in the early years and these resources are a response to the rapid integration of tablets in primary schools.

One notable research-led example is ScratchJr, a graphical programing language based on Scratch that has been redesigned for children aged between five and seven

years by capitalising on the interactions afforded by tablets. According to Flannery et al. (2013), the simplification of processes in ScratchJr does not hide important conceptual information (such as displaying programmed steps) in the same way as tools such as floor robots; instead, the design offers the exploratory power of an environment like Scratch, but simplifies the complexity by removing cognitive demands on, for instance, reading ability.

Tangible designs that use physical interfaces rather than a screen are becoming more widespread for the early years (Manches & O'Malley, 2012). Popular classroom resources include floor robots, such as Beebot, BigTrack and Pixie), that represent iterations of the original Logo floor turtle of almost thirty years earlier but more innovative approaches are beginning to emerge. Cubetto, for instance, uses blocks placed in a board featuring cut-out shapes, and Dash and Dot are toy robots that enable children from four years old to explore the effects of inputting simple instructions. Robot Turtles is a non-digital board game with playing cards that teaches the fundamentals of programming to children from the age of three by providing a social context in which to create instructions for navigating a turtle around a board.

Programming tools such as these do not make the same interaction demands as screen-based products and should be easier for young children to manipulate and understand, but they also raise questions. Which tools are most beneficial and at what stage of development? How do they support different programming concepts? To what extent do skills from one tool transfer to others or to broader ideas within and beyond computing? Now that they are marketed to parents as well as teachers, what is the role of the adult in using these tools with children? By demonstrating the potential to simplify the procedural steps of programming, tools such as ScratchJr raise questions about what, if any, conceptual challenges young children face.

The TangibleK project represents a notable effort to address such questions by developing an early years curriculum in robotics (Bers, 2010). Robotics, engineering and design process, flow, loops and parameters, sensors and branches are identified by Bers as powerful ideas for early learning. Many of these terms may appear challenging for practitioners but Bers illustrates their relationship with more familiar ideas in the early years, from ‘cause and effect’ and ‘storytelling’ to ‘number sense’ and ‘scientific observations’, relating computing concepts to core cogntive skills such as sequencing. A demonstration of how young children’s activities with robots transferred to improvements in story sequencing (Kazakoff, Sullivan, & Bers, 2013), for instance, suggests that these skills are not simply prerequisites for computing activities but might be developed through them. While there have been praiseworthy efforts to consider the developmental steps involved in acquiring computing concepts (e.g. by the Computing at School community), the research base for these assertions is limited.

Research can make a contribution to understanding how early years computing activities relate to other learning, from cognitive skills such as self-regulation or planning to scientific experimentation and communication skills. Where research is tool-specific, it may obscure the extent to which benefits can be replicated with other resources, although Roland’s (2013) work examining young children’s ability to generate algorithmic solutions to everyday problems such as sorting pencils of different lengths provides an exception.

We are beginning to amass more evidence for the potential of programming tools to help children develop computational thinking but the role of the educator in

encouraging reflection and bridging ideas has not yet been fully recognised. Without this, children's attention may be overly focused on creating the right coding procedures, rather than understanding the broader concepts of their activity.

**Summary**

This paper suggests that further consideration of computing education for children in the early years is timely. An increasing number of resources that capitalise on the affordances of new technologies such as tablet devices, alongside new curricula across the age range, demonstrate the opportunities. Yet there is a danger that the pace of change has mitigated against sufficient time to develop a research-informed pedagogy. It is challenging to develop an evidence-based approach to teaching computing to younger children in an environment in which educational tools evolve rapidly and teachers feel under-prepared but, without it, there is a risk of demotivating children in an important area of the curriculum.

Consideration of a wider body of research literature may offer a partial solution to the lack of recent pedagogical research in this domain and this discussion identifies three areas which may have particular value: prior research on Logo, more recent work on computational thinking, and the introduction of STEM education in the early years. The first of these provides an historical perspective on previous attempts to integrate computing into education, and the importance of considering the role of the teacher. The second allows us to link this prior work to more recent efforts to understand the broader thinking skills behind computing, and how this helps clarify the relevance to other areas of learning for younger children.

In order to move this debate forward, we need to draw upon a range of literature, both past and present, to inform computing education in the early years. As Rushby (2008, p.195) states in his editorial of a special issue of *British Journal of Educational Technology*, "We are not very good at maintaining our community memory of what has gone before and so we are condemned to reject history". An initial challenge for the field has been to use the notion of computational thinking to underpin meaningful definitions of computing. This emphasis on the broader scope of computing enables practitioners to draw upon their knowledge, confidence and ability to link with other ideas and activities. Programming plays a key role, but interpreting it in the wider sense of problem solving makes it easier to clarify the relationships between activities, whether this is giving someone directions or programming a floor robot.

The increasing number of tools available to help educators teach computing is to be welcomed. Yet there is work to do in terms of evaluating how easily teachers can integrate these resources into early years pedagogy, especially as technology is often discussed as a separate topic in curriculum documentation. Many educators work from the premise that integrating technology into learning in early years settings requires more teacherly instruction than they feel comfortable with and they may see it as undermining the more child-centered notion of early years education that is prevalent in many countries. It is important, then, to address questions such as the extent to which the tools allow teachers to address the unique social and emotional context of their classroom and how easily the resources can be adapted to support different ideas and activities within such a dynamic setting. Teachers' professional development needs to balance a desire to promote reflection on the role of computing in children's everyday lives, including the need to foster thinking skills, with recognition of the need to provide non-specialist teachers with off-the-shelf tools and

activities to address specific curriculum objectives, recognising that this is about more than developing competence in orchestrating children's step-by-step activity with particular tools. In this regard, there is much to be gained from encouraging researchers to engage more fully with organisations such as Computing at School and educational events such as BETT to build research-informed practitioner communities.

Research on education in the early years has moved forward in the last few decades, informed by an understanding of the multimodality of young children's learning, as well as socio-political changes that emphasise the need to respect young children's views. As a result, it has been increasingly important to encourage children to reflect upon the world around them and to be engaged in real world problems and solutions. As computational devices, and the algorithms that drive them, become more pervasive in children's lives, it becomes necessary to empower all children with the understanding and confidence not only to navigate their environment but to shape it. Newly available products have highlighted the potential to involve ever-younger children and while this presents an opportunity, it also presents a risk that an under-informed approach not only disempowers teachers but also demotivates children. This paper calls for debate on the role of computing in the early years in order to realise the opportunities presented by bridges between research and practice and to enable us to move forward, informed by the past.

## References


Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community? *ACM Inroads, 2*(1), 48-54.

Bers, M. U. (2010). The TangibleK Robotics Program: Applied Computational Thinking for Young Children. *Early Childhood Research & Practice, 12*(2), n2.

Brown, N. C., Sentance, S., Crick, T., & Humphreys, S. (2014). Restart: The resurgence of computer science in UK schools. *ACM Transactions on Computing Education (TOCE), 14*(2), 9.

Cheung, R. H. P. (2013). Exploring the use of the pedagogical framework for creative practice in preschool settings: A phenomenological approach. *Thinking Skills and Creativity, 10*, 133-142.

Clements, D. H., & Gullo, D. F. (1984). Effects of computer programming on young children's cognition. *Journal of Educational Psychology, 76*(6), 1051-1058.

Compton, A., Glauert, E., Stylianido, F., Craft, A., Cremin, T., & Havu-Nuutinen, S. (2014). Creative Little Scientist: Set of Recommendations to Policy Makers and Stakeholders: Executive Summary. In s. Compton, E. Glauert, F. Stylianido, E. Agogi, Craft, T. Cremin, & S. Havu-Nuutinen (Eds.). Ellinogermaniki Agogi.

Cooper, S., Bookey, L., & GruenBaum, P. (2014). Future Directions in Computing Education Summit Part One: Important Computing Education Research Questions.

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5-6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education, 63*, 87-97.

Flannery, L. P., Silverman, B., Kazakoff, E. R., Bers, M. U., Bontá, P., & Resnick, M. (2013). *Designing ScratchJr: support for early childhood learning through computer programming*. Paper presented at the Proceedings of the 12th International Conference on Interaction Design and Children.

Grover, S., & Pea, R. (2013). Computational Thinking in K–12 A Review of the State of the Field. *Educational Researcher, 42*(1), 38-43.

Kazakoff, E. R., Sullivan, A., & Bers, M. U. (2013). The Effect of a Classroom-Based Intensive Robotics and Programming Workshop on Sequencing Ability in Early Childhood. *Early Childhood Education Journal*, 1-11.

Layman, J., & Hall, W. (1988). Logo: A cause for concern. *Computers & Education, 12*(1), 107-112.

Manches, A. (2013). Evaluating the Learning Benefits of New Forms of Interaction. *Handbook of Design in Educational Technology*, 425.

Manches, A., & O'Malley, C. (2012). Tangibles for learning: a representational analysis of physical manipulation. *Personal and Ubiquitous Computing, 16*(4), 405-419.

Papert, S. (1987). *A critique of technocentrism in thinking about the school of the future*. Epistemology and Learning Memo #2. MIT.

Pea, R. D. (1983). Logo programming and problem solving. In E.Scanlon & T. O'Shea (Eds.), *Educational Computing* (pp. 155-160). Chichester: Wiley.

Perlman, R. (1976). Using computer technology to provide a creative learning environment for preschool children.

Plowman, L., & Stephen, C. (2005). Children, play, and computers in pre-school education. *British Journal of Educational Technology, 36*(2), 145-157.

Rittle-Johnson, B., Siegler, R. S., & Alibali, M. W. (1999). Conceptual and procedural knowledge of mathematics: Does one lead to the other? *Journal of Educational Psychology, 91*(1), 175-189.

Robertson, J. (2013). The influence of a game-making project on male and female learners' attitudes to computing. *Computer Science Education, 23*(1), 58-83.

Roland, T. (2013). Algorithmics for Preschoolers—A Contradiction? *Creative Education, 4*, 557-562.

Royal Society. (2012). Shut down or restart: The way forward for computing in UK schools. In S.Furber (Ed.). London.

Sargent, R., Resnick, M., Martin, F., & Silverman, B. (1996). *Building and learning with programmable bricks*. New Jersey: Laurence Erlbaum Associates.

Sutherland, R. (1993). Connecting theory and practice: Results from the teaching of Logo. *Educational studies in mathematics, 24*(1), 95-113.

Sylva, K., Melhuish, E., Sammons, P., Siraj-Blatchford, I., & Taggart, B. (Eds.). (2009). *Early childhood matters: evidence from the effective pre-school and primary education project*. London: Routledge.

Wajcman, J. (2007). From women and technology to gendered technoscience. *Information, Community and Society, 10*(3), 287-298.

Wilson, C., Sudol, L. A., Stephenson, C., & Stehlik, M. (2010). Running on empty: The failure to teach K-12 computer science in the digital age. Association for Computing Machinery. *Computer Science Teachers Association*.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366*(1881), 3717-3725.

Wing, J. M. (2011). *Computational Thinking: What and Why?* OurCS Workshop Presentation. Carnegie Mellon University.

Wyeth, P., & Wyeth, G. (2001). *Electronic Blocks: tangible programming elements for preschoolers.* Paper presented at the Proceedings of the Eighth IFIP TC13 Conference on Human-Computer Interaction, Amsterdam.

**Resources referred to above** (in alphabetical order):

All links were live on 7th June 2015.

Barefoot project: http://barefootcas.org.uk

Bee-Bot: www.bee-bot.us

Code.org: http://code.org

Code Club: www.codeclub.org.uk

CoderDojo: https://coderdojo.com

Computing at School: http://www.computingatschool.org.uk/

Cubetto: http://www.primo.io

Dash and Dot: https://www.makewonder.com

Hopscotch: http://www.gethopscotch.com

iCompute: http://www.icompute-uk.com/news/computing-in-the-foundation-stage/

Kodable: https://www.kodable.com

Lightbot: http://lightbot.com/hocflash.html

Pixie: http://www.swallow.co.uk/pixie/pixie1.htm

Robot Turtles: www.robotturtles.com

Scratch: scratch.mit.edu

ScratchJr: http://www.scratchjr.org

Tynker: http://www.tynker.com

[1] Education in the United Kingdom is devolved to the individual administrations of England, Scotland, Wales and Northern Ireland and they do not share a common curriculum. We focus on England because political initiatives have led to accelerated changes in computing education there.